



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



Học tăng cường

Nguyễn Phi Lê



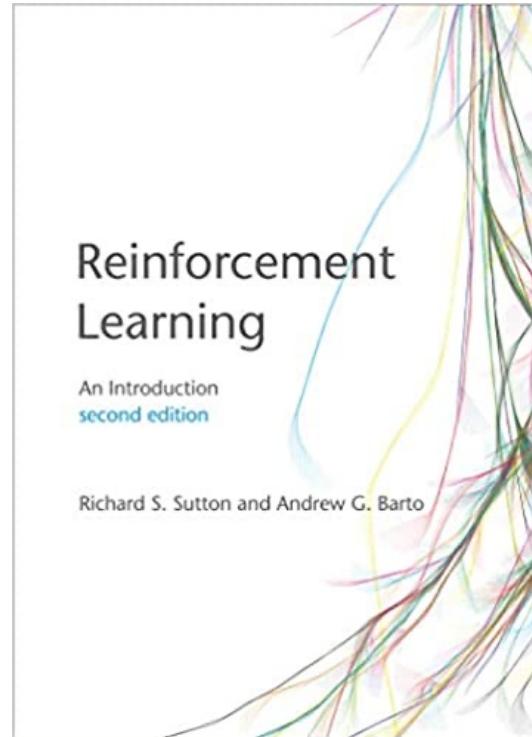
Mục lục

- Reinforcement learning (RL)
 - Nguyên lý
 - Đánh giá độ tốt của action
 - Chiến lược lựa chọn action
 - Tabular RL
 - Approximate RL
- Deep Reinforcement learning (DRL)
 - Nguyên lý và luồng hoạt động
 - Một số mô hình DRL phổ biến

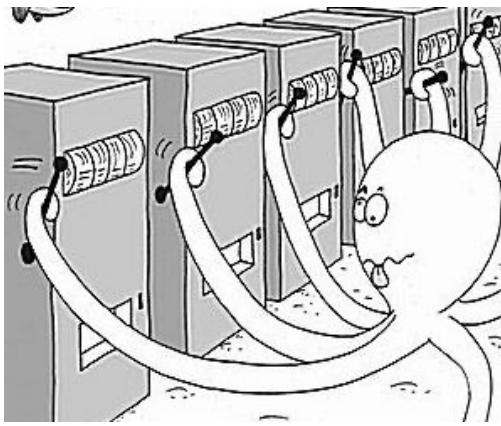
Reinforcement learning

Tài liệu tham khảo

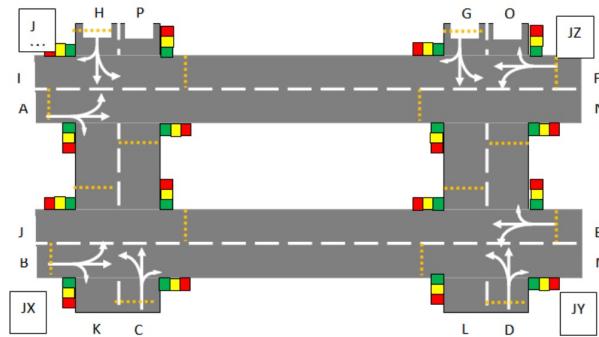
- Richard S. Sutton and Andrew G. Barto,
“Reinforcement Learning: An Introduction”, MIT
Press



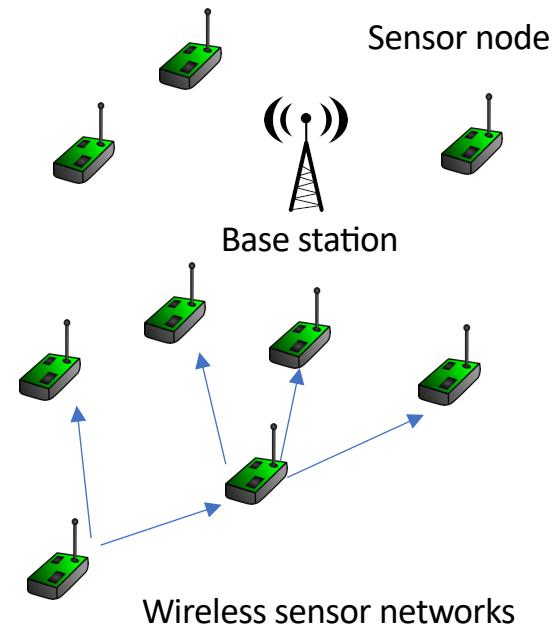
Interactive learning



Multi-Armed Bandit



Traffic light control



Wireless sensor networks

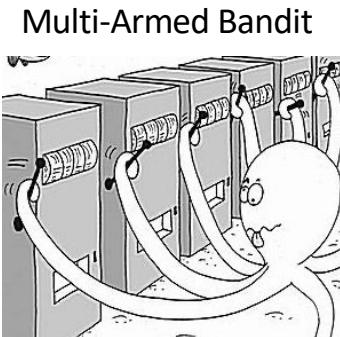
- ✓ Chủ thể không biết hành động nào là tối ưu → trial-and-error
- ✓ Một hành động không phải chỉ ảnh hưởng tới hiện tại mà có thể ảnh hưởng tới cả tương lai → delayed reward

Định nghĩa

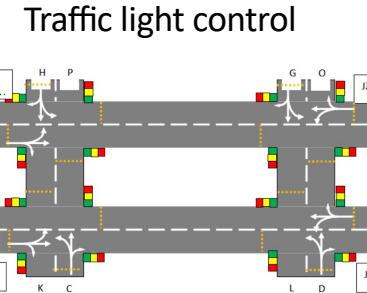
- Học tăng cường (Reinforcement learning)
 - Học hành động (học việc ra quyết định) — làm sao để ánh xạ trạng thái hiện tại với hành động nên thực hiện — nhằm đạt được mục tiêu (goal)
- Reward hypothesis
 - All of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward) (Richard S. Sutton, RL: An introduction, 2018)

Mục tiêu (Goal) và phần thưởng (reward)

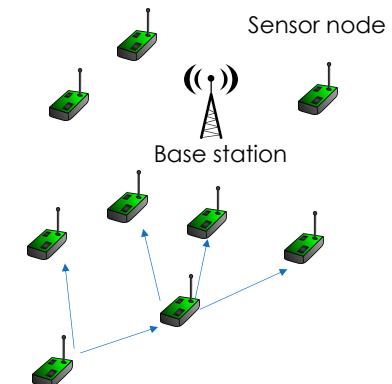
- Phần thưởng: là một đại lượng vô hướng nhận được sau mỗi hành động
- Mục tiêu: tối ưu hoá tổng phần thưởng luỹ tích trong một thời gian dài



Goal = tối đa hoá tổng lợi nhuận
Reward = lợi nhuận thu được sau mỗi lần chơi



Goal = giảm tắc nghẽn
Reward = waiting time, queue length, lane speed, ...



Goal = tăng thời gian sống của mạng
Reward = load balancing, route length, ...

RL vs các kỹ thuật học máy khác

- RL vs học có giám sát

- Học có giám sát: học từ một tập dữ liệu được gán nhãn trước.

- Biết hành động tối ưu là gì

- Reinforcement learning: không biết hành động tối ưu là gì

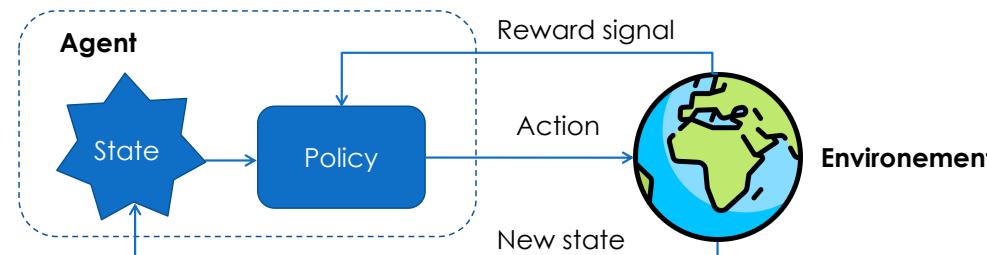
- RL vs học không giám sát

- Học không giám sát: tìm ra cấu trúc ẩn của một tập dữ liệu không gán nhãn

- Reinforcement learning: tối ưu hóa tổng luỹ tích của phần thưởng

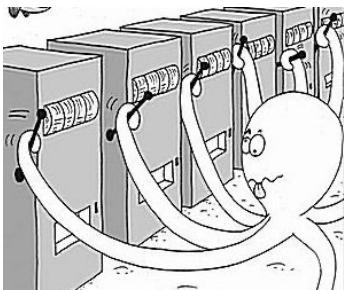
RL framework

- Agent (Chủ thể): người đưa ra quyết định và thực hiện hành động
- Action (Hành động): là hành động được thực hiện bởi chủ thể
- State (trạng thái): có thể là bất cứ cái gì, miễn là ta có thể dựa trên đó để đưa ra quyết định
- Policy (chiến lược): một ánh xạ từ trạng thái sang hành động
 - Có thể là ngẫu nhiên: định nghĩa bằng xác suất
- Reward (phần thưởng): phản ánh mục tiêu của bài toán RL
 - Mục tiêu của bài toán RL là cực đại hóa tổng phần thưởng trong quãng thời gian dài
 - reward: là phần thưởng tức thời, sau khi thực hiện 1 hành động
- Value (giá trị): là tổng luỹ kế của phần thưởng từ thời điểm hiện tại tới toàn bộ tương lai sau này
 - Reward: thể hiện thông tin ở thời điểm hiện tại
 - Value: thể hiện thông tin trong quãng thời gian dài



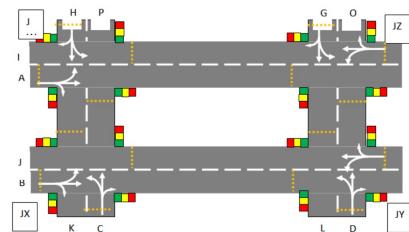
RL framework

Multi-Armed Bandit

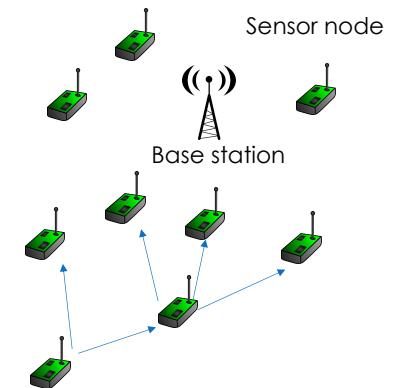


Goal = tối đa hóa tổng lợi nhuận
Reward = lợi nhuận thu được sau mỗi lần chơi
Agent: robot
Action: ID của máy sẽ chơi

Traffic light control



Goal = giảm tắc nghẽn
Reward = waiting time, queue length, lane speed, ...
Agent: **các** đèn giao thông (hoặc **một** chương trình điều khiển trung tâm)
Action: xanh/đỏ/vàng



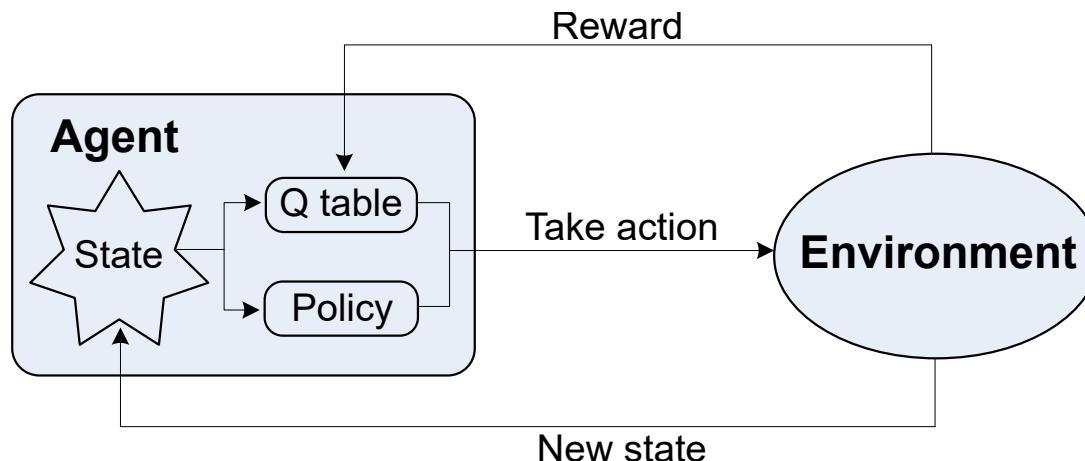
Goal = tăng thời gian sống của mạng
Reward = load balancing, route length, ...
Agent: **các** sensor (hoặc **một** chương trình điều khiển trung tâm)
Action: nút tiếp theo sẽ nhận gói tin

Mục lục

- Reinforcement learning (RL)
 - Nguyên lý
 - Đánh giá độ tốt của action
 - Chiến lược lựa chọn action
 - Tabular RL
 - Approximate RL
- Deep Reinforcement learning (DRL)
 - Nguyên lý và luồng hoạt động
 - Một số mô hình DRL phổ biến

Nguyên lý của RL

- Mục tiêu của học tăng cường: đưa ra quyết định
 - Input: trạng thái hiện tại
 - Output: hành động



Các kiểu mô hình RL

- Value-based
- Policy-based

Các kiểu mô hình RL

- Value-based
- Policy-based

Nguyên lý của RL

- Mục tiêu của học tăng cường: đưa ra quyết định
 - Input: trạng thái hiện tại
 - Output: hành động
 - Cơ sở đưa ra quyết định
 - Độ tốt của action
 - Chiến lược lựa chọn action

Nguyên lý của RL

- Mục tiêu của học tăng cường: đưa ra quyết định
 - Input: trạng thái hiện tại
 - Output: hành động
 - Cơ sở đưa ra quyết định
 - Độ tốt của action
 - Làm sao đánh giá được độ tốt của action?
 - Chiến lược lựa chọn action

Nguyên lý của RL

- Mục tiêu của học tăng cường: đưa ra quyết định
 - Input: trạng thái hiện tại
 - Output: hành động
 - Cơ sở đưa ra quyết định
 - Độ tốt của action
 - Làm sao đánh giá được độ tốt của action?
 - Ước lượng độ tốt của action dựa trên kinh nghiệm thực hiện hành động đấy trong quá khứ
 - Chiến lược lựa chọn action

Nguyên lý của RL

- Mục tiêu của học tăng cường: đưa ra quyết định
 - Input: trạng thái hiện tại
 - Output: hành động
 - Cơ sở đưa ra quyết định
 - Độ tốt của action
 - Làm sao đánh giá được độ tốt của action?
 - Ước lượng độ tốt của action dựa trên kinh nghiệm thực hiện hành động đấy trong quá khứ
 - Lượng hoá thông qua hàm action-value
 - Chiến lược lựa chọn action

Nguyên lý của RL

- Mục tiêu của học tăng cường: đưa ra quyết định
 - Input: trạng thái hiện tại
 - Output: hành động
 - Cơ sở đưa ra quyết định
 - Độ tốt của action
 - Làm sao đánh giá được độ tốt của action?
 - Ước lượng độ tốt của action dựa trên kinh nghiệm thực hiện hành động đấy trong quá khứ
 - Lượng hoá thông qua hàm action-value
 - Chiến lược lựa chọn action
 - Dựa trên thông tin về độ tốt của action
 - Chiến thuật exploration-exploitation
 - Kết hợp các giải thuật heuristic

Nguyên lý của RL

- Mục tiêu của học tăng cường: đưa ra quyết định
 - Input: trạng thái hiện tại
 - Output: hành động
 - Cơ sở đưa ra quyết định
 - Độ tốt của action
 - Làm sao đánh giá được độ tốt của action?
 - Ước lượng độ tốt của action dựa trên kinh nghiệm thực hiện hành động đấy trong quá khứ
 - Lượng hoá thông qua hàm action-value
 - Chiến lược lựa chọn action
 - Dựa trên thông tin về độ tốt của action
 - Chiến thuật exploration-exploitation
 - Kết hợp các giải thuật heuristic

Mục lục

- RL
 - Nguyên lý
 - Đánh giá độ tốt của action
 - Chiến lược lựa chọn action
 - Tabular RL
 - Approximate RL
- DRL
 - Nguyên lý và luồng hoạt động
 - Một số mô hình DRL phổ biến

Hàm value

- Hàm State value: Đánh giá độ tốt của trạng thái s
 - Là giá trị kỳ vọng khi bắt đầu tại trạng thái s và thực hiện chiến lược π từ đây về sau
- Hàm Action value: Đánh giá độ tốt của hành động a tại một trạng thái s
 - Là giá trị kỳ vọng khi bắt đầu từ trạng thái s , thực hiện hành động a , và thực hiện chiến lược π từ đây về sau

Hàm State value cho chiến lược π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s]$$

Hàm Action value cho chiến lược π

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$$

Đánh giá độ tốt của action

- → Tiêu chí đánh giá độ tốt của hành động?
- → Thế nào là 1 hành động tốt nhất tại mỗi thời điểm t ?

long-term accumulative reward

- Maximize:

$$\bullet G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n R_{t+n-1} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Mục tiêu tối ưu

Reward nhận được khi thực hiện action a tại thời điểm t

Reward nhận được nếu tuân theo policy π từ thời điểm $t + 1$ trở đi

Đánh giá độ tốt của action

- $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n R_{t+n-1} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- Hàm action-value $q_{\pi}(s, a)$: **độ tốt** của việc thực hiện hành động a khi ở trạng thái s
- $q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$: **giá trị kỳ vọng** của G_t khi thực hiện hành động a tại trạng thái s , và từ đấy về sau tuân theo chiến lược π
 - Giá trị kỳ vọng = trung bình cộng của **tất cả các lần** thực hiện hành động a tại trạng thái $s \rightarrow$ phải thực hiện rất nhiều lần
- Đánh giá độ tốt của action = **ước lượng** hàm $q_{\pi}(s, a)$
 - Điểm mấu chốt trong việc đánh giá độ tốt của action
 - Cốt lõi của các thuật toán RL

Đánh giá độ tốt của action

- Ý tưởng chính của các thuật toán RL

- Thay thế “giá trị kỳ vọng” bằng “ước lượng dựa trên mẫu” (Replacing expected value by sample-based estimation)
 - Mỗi lần thực hiện xong 1 action → có 1 mẫu → dùng mẫu đấy để ước lượng

Lý tưởng: $q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$

Thực tế: mỗi lần thực hiện xong 1 action

→ sử dụng thông tin nhận được để ước lượng $\mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$

Phương pháp ước lượng (iterative estimation):

- Khởi tạo $q_{\pi}(s, a)$ bằng 1 giá trị bất kỳ
- Update $q_{\pi}(s, a)$ sau mỗi lần thực hiện action
 - ❖ $q_{\pi}(s, a)$ hiện tại
 - ❖ Giá trị ước lượng của $\mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$

Làm sao để ước lượng
 $\widehat{q}_{\pi}(s, a)$?

$$\rightarrow \text{New } q_{\pi}(s, a) = (1 - \alpha) \text{ current } q_{\pi}(s, a) + \alpha \times \widehat{q}_{\pi}(s, a)$$

Làm thế nào để ước lượng $q_\pi(s, a)$

Công thức Bellman

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi[R_{t+1} + \sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, a') | S_t = s, A_t = a]$$

Action value của hành động a , tại trạng thái s

Kỳ vọng

Reward nhận được ngay sau khi thực hiện action a

Độ tốt của các hành động ở trạng thái tiếp theo

Làm thế nào để ước lượng $q_\pi(s, a)$

Công thức Bellman

$$\bullet q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, a') | S_t = s, A_t = a]$$

Đối tượng cần ước lượng: $\widehat{q}_\pi(s, a)$

$$\text{New } q_\pi(s, a) = (1 - \alpha) \text{ current } q_\pi(s, a) + \alpha \times \widehat{q}_\pi(s, a)$$
$$Q(s, a) = (1 - \alpha) Q(s, a) + \alpha (r + \gamma Q(s', a')) \quad \text{SARSA}$$

s' : trạng thái chuyển đến sau khi thực hiện hành động a
 a' : hành động tại trạng thái s'

Làm thế nào để ước lượng $q_\pi(s, a)$

- Công thức Bellman
- $q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, a') | S_t = s, A_t = a]$
- $q_{\pi^*}(s, a) = \mathbb{E}_{\pi^*}[R_{t+1} + \gamma q_{\pi^*}(S_{t+1}, a') | S_t = s, A_t = a]$

$$\sim \mathbb{E}_{\pi^*} \left[R_{t+1} + \gamma \max_{\pi} q_\pi(S_{t+1}, a') | S_t = s, A_t = a \right]$$

Ước lượng: $\widehat{q}_{\pi^*}(s, a) = r + \gamma \max_{a'} Q(s', a')$

New $q_\pi(s, a) = (1 - \alpha) \text{ old } q_\pi(s, a) + \alpha \times \widehat{q}_{\pi^*}(s, a)$

Q-learning $Q(s, a) = (1 - \alpha) Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') \right)$

SARSA $Q(s, a) = (1 - \alpha) Q(s, a) + \alpha (r + \gamma Q(s', a'))$

Làm thế nào để ước lượng $q_\pi(s, a)$

Công thức Bellman cho state-value function

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | S_t = s] \end{aligned}$$

Monte Carlo

$$= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$\text{New } v_\pi(s) = (1 - \alpha) \text{ current } v_\pi(s) + \alpha \times E$$

$$V(S_t) = (1 - \alpha)V(S_t) + \alpha G_t$$

Temporal
Difference
(TD)

$$V(S_t) = (1 - \alpha)V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}))$$

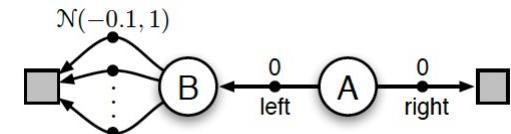
Double Q-learning

- Double Q-learning [NIPS 2010]
- Hiện tượng overestimation của Q-learning

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') \right)$$

Action tốt nhất ở trạng thái tiếp theo

- $\max_{a'} Q(s', a')$ được dùng như 1 approximation của $\mathbb{E}_{\pi^*}[R_{t+1} + \gamma q_{\pi^*}(S_{t+1}, a')]|S_t = s, A_t = a]$



Overestimation:
Kỳ vọng của action "left" là -0.1
Maximum của action "left" là 1

Vai trò của Q_1 và Q_2 thay đổi luân phiên (có thể ngẫu nhiên)

$$Q_1(s, a) = (1 - \alpha)Q_1(s, a) + \alpha \left(r + \gamma Q_2 \left(s', \arg \max_{a'} Q_1(s', a') \right) \right)$$

$$Q_2(s, a) = (1 - \alpha)Q_2(s, a) + \alpha \left(r + \gamma Q_1 \left(s', \arg \max_{a'} Q_2(s', a') \right) \right)$$

Overestimation: hiện tượng approximation quá lớn so với giá trị đúng

- Double Q-learning: sử dụng 2 Q functions
- Một Q function dùng để chọn action tốt nhất
 - Hàm này chọn ra action tạo ra $\max_{a'} Q(s', a')$
 - Một Q function dùng để estimate Q value
 - Dùng 1 hàm khác để tính $\max_{a'} Q(s', a')$ khi update Q-value

Câu hỏi

- Mục tiêu của RL là gì?
- Goal và reward khác nhau như thế nào?
- Value-based RL là các thuật toán như thế nào?
- Điểm khác nhau giữa Q-learning và SARSA là gì?

Mục lục

- RL
 - Nguyên lý
 - Đánh giá độ tốt của action
 - Chiến lược lựa chọn action
 - Tabular RL
 - Approximate RL
- DRL
 - Nguyên lý và luồng hoạt động
 - Một số mô hình DRL phổ biến

Làm thế nào để lựa chọn action?

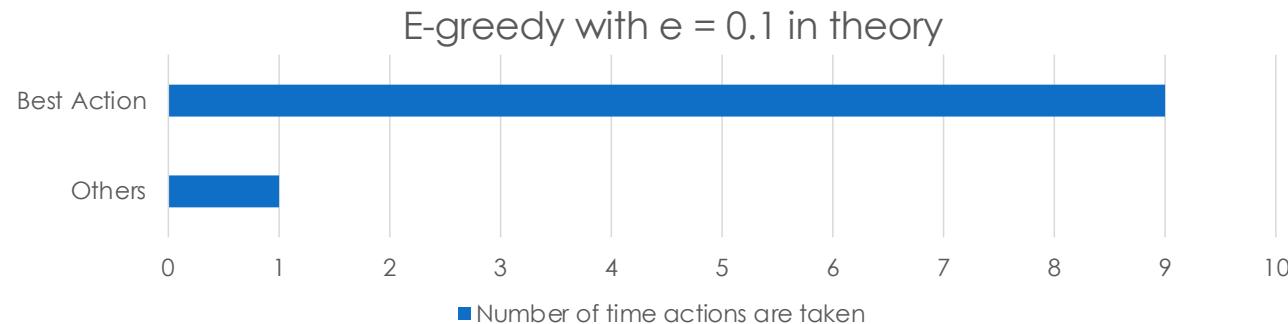
- Dựa vào $Q(s, a)$
 - Tại mỗi thời điểm t (tương ứng với state s), luôn lựa chọn action a có $Q(s, a)$ lớn nhất ← **exploitation**
 - Vấn đề: cực đại địa phương → cần khám những hành động chưa bao giờ thực hiện ← **exploration**
 - Các chiến thuật lựa chọn action cơ bản
 - ϵ -greedy
 - Time adaptive Epsilon (annealing ϵ)
 - ϵ -soft
 - Value adaptive Epsilon
 - Sigmoid Epsilon

Làm thế nào để lựa chọn action?

- ϵ -greedy: cố định tham số $\epsilon \in (0,1)$
 - Xác suất ϵ : chọn action ngẫu nhiên trong tập các action
 - Xác suất $1 - \epsilon$: chọn action có action-value (Q value) lớn nhất
- Time adaptive epsilon (annealing ϵ)
 - $\epsilon = \frac{1}{\log(time+c)}$
 - c : hằng số dương rất nhỏ; $time$: số lần đã thực hiện các action
 - Càng thực hiện nhiều thì càng khám phá ít
- ϵ -soft
 - Đảm bảo xác suất lựa chọn một action bất kỳ đều không nhỏ hơn $\frac{\epsilon}{|A_s|}$
 - $|A_s|$: số lượng actions

ϵ -greedy

- Định nghĩa số dương ϵ nhỏ ($\epsilon = 0.01$)
- Tại mỗi thời điểm, agent ra quyết định như sau
 - Xác định số $p \in (0,1)$
 - $A_t = \begin{cases} argmax_a(q(s, a)), if \ p \geq \epsilon \\ random\ action, otherwise \end{cases}$



ϵ -greedy multi-armed bandit

- Action space: $A = \{a_1, \dots, a_n\}$

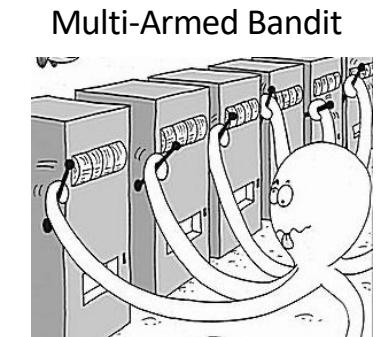
- Reward: lợi nhuận thu được

- Hàm Value

- $Q_t(a) \doteq \frac{\text{sum of rewards when action } a \text{ taken prior to } t}{\text{number of times action } a \text{ taken prior to } t}$

- Exploration strategy

- ϵ -greedy



Multi-Armed Bandit
Goal = maximizing total gain
Reward = gain at every turn

A simple bandit algorithm

Initialize, for $a = 1$ to k :

$$Q(a) \leftarrow 0$$
$$N(a) \leftarrow 0$$

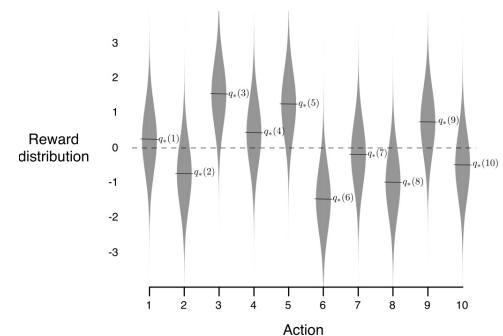
Loop forever:

$$A \leftarrow \begin{cases} \arg \max_a Q(a) & \text{with probability } 1 - \epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases} \quad (\text{breaking ties randomly})$$

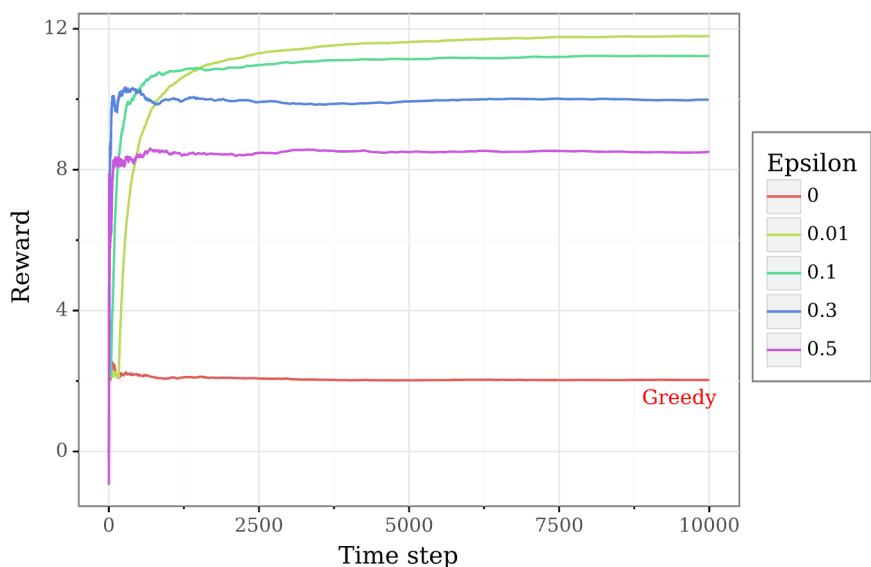
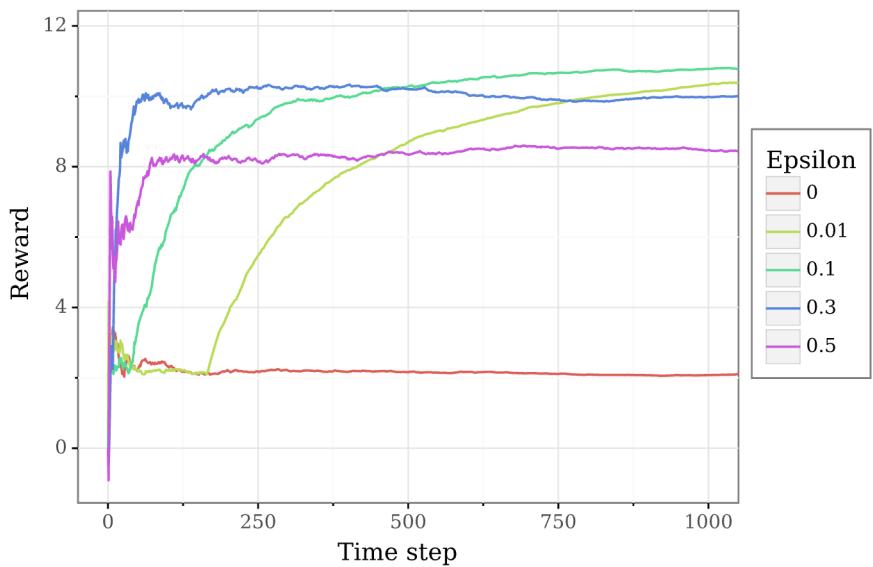
$$R \leftarrow \text{bandit}(A)$$

$$N(A) \leftarrow N(A) + 1$$

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$$

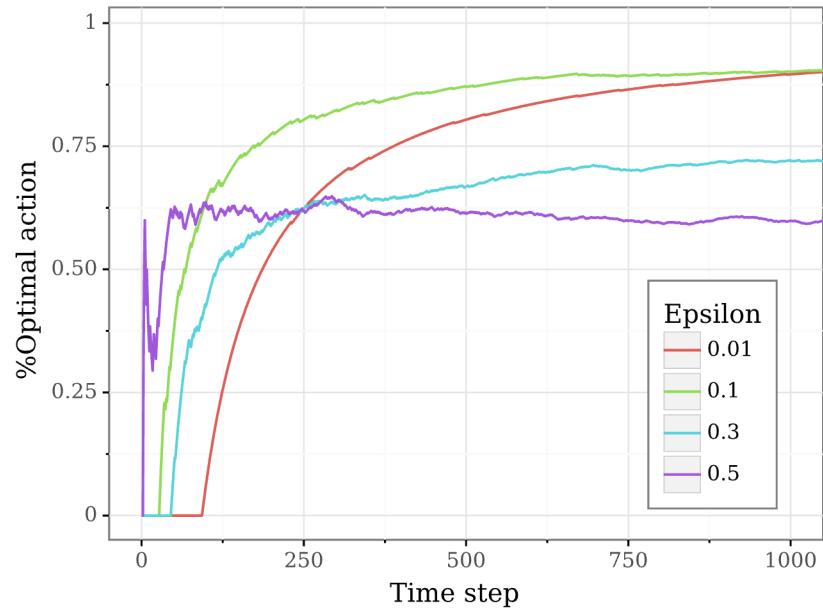
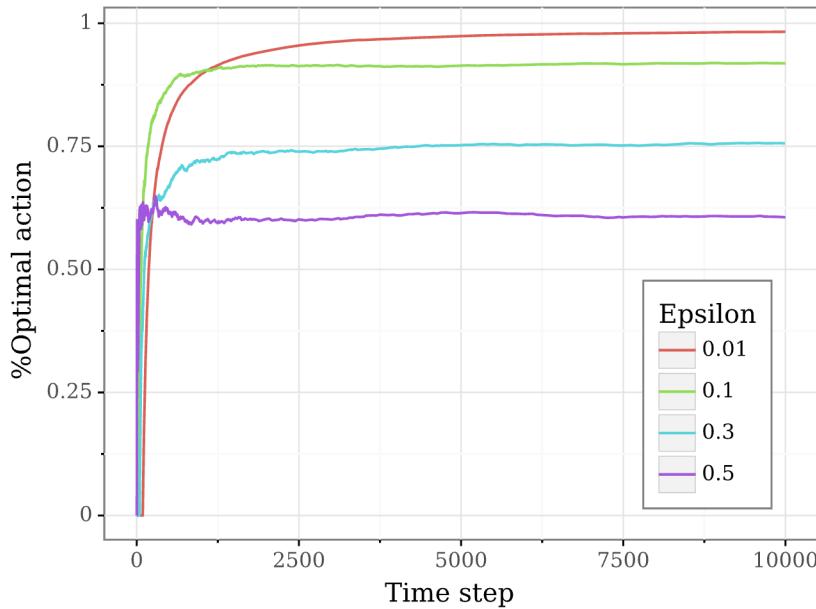


Exploration vs Exploitation



ϵ càng lớn, khám phá càng nhiều
→ Reward càng kém ổn định

Exploration vs Exploitation



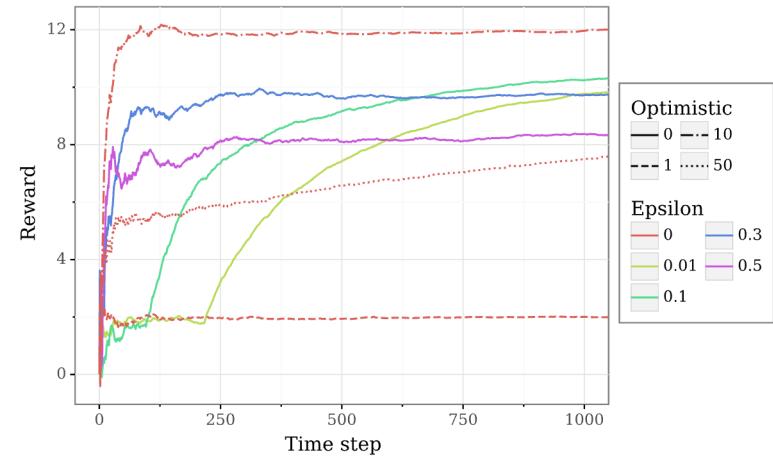
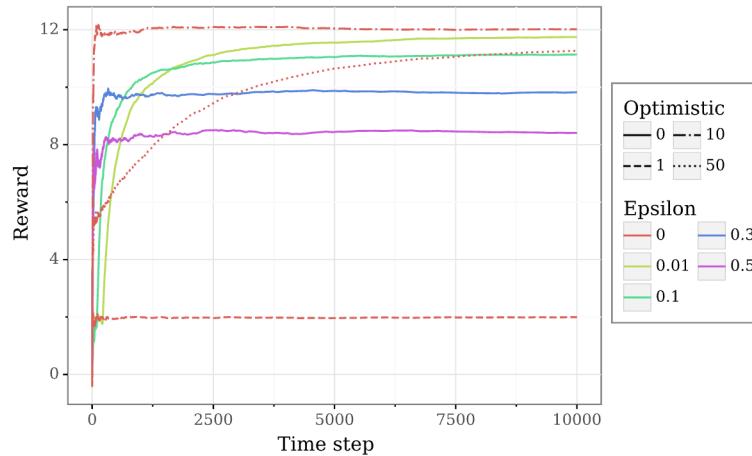
Tần suất chọn hành động tối ưu tỉ lệ nghịch với độ lớn của ϵ

Khởi tạo lạc quan (Optimistic initialization)

- Khởi tạo $Q_0(a) = X \forall a$ (X là số rất lớn)
 - $Q_t(a) \doteq \frac{\text{sum of rewards when action } a \text{ taken prior to } t}{\text{number of times action } a \text{ taken prior to } t}$
- Chiến thuật lựa chọn hành động
 - Tại mỗi step t : choose $A_t = \operatorname{argmax}_a(Q_t(a))$
 - Mỗi hành động đều được chọn ít nhất 1 lần
 - $Q_1(a) = \frac{Q_0(a) + R_1}{2} < Q_0(a)$

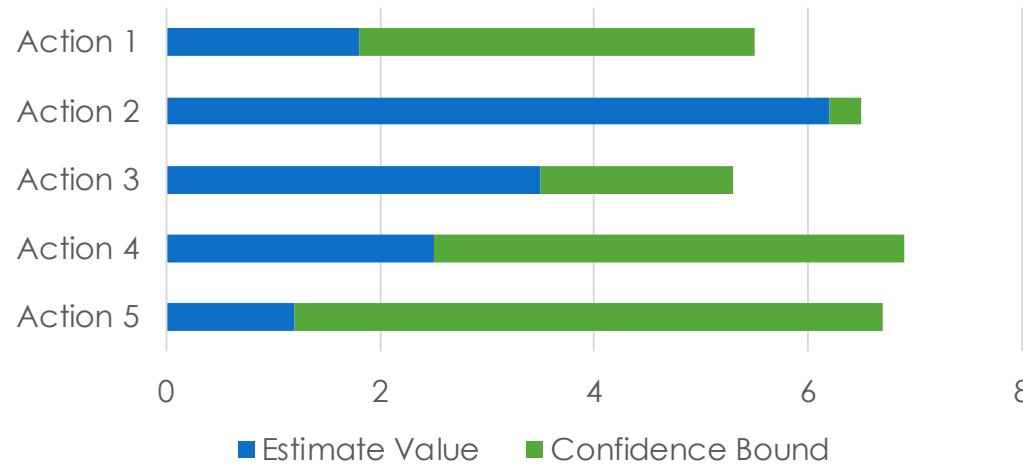
Khởi tạo lạc quan (Optimistic initialization)

- Khám phá rất nhiều ban đầu
 - Càng về sau càng giảm, cuối cùng thì giống greedy
- Giá trị của X ảnh hưởng lớn tới hiệu năng thuật toán
 - Cần chọn X hợp lý



Upper confidence bound (UCB)

- Hành động nào đã thực hiện nhiều thì nên khám phá ít lại
- Confidence bound được định nghĩa dựa trên số lần một hành động không được chọn
- Ưu tiên các hành động
 - Giá trị ước lượng (value) cao → exploitation
 - Confidence bound lớn → exploration



Upper confidence bound (UCB)

- Mô hình toán học

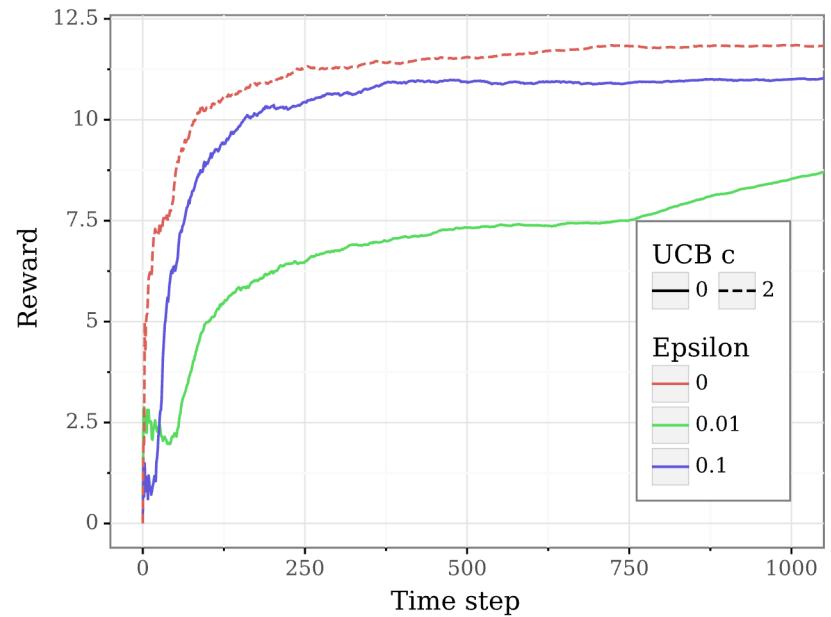
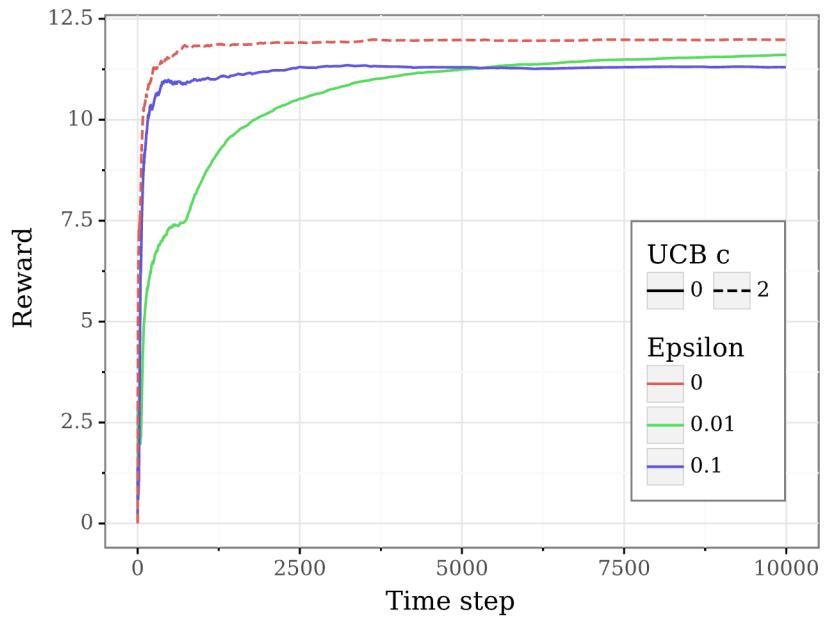
- Số lần 1 hành động a được chọn là: $N_t(a)$
- Tổng số timestep cho tới thời điểm hiện tại: N_t
- Confidence bound

- $CB_t(a) = c \sqrt{\frac{N_t}{N_t(a)}}$ (c : a constant)

- Chiến thuật lựa chọn hành động

- Tại bước thứ t : choose $A_t = argmax_a \left(Q_t(a) + c \sqrt{\frac{N_t}{N_t(a)}} \right)$
- Ý nghĩa
 - Nếu a ít được chọn $\rightarrow N_t(a)$ giảm $\rightarrow CB_t(a)$ tăng \rightarrow sẽ được ưu tiên thực hiện trong các lần tới

Upper confidence bound (UCB)



Một số chiến thuật khám phá khác

- Value adaptive Epsilon
 - Ý tưởng: điều chỉnh exploration dựa trên sự thay đổi của value function
 - Nên explore nhiều khi agent CHƯA biết rõ về môi trường (khi value function thay đổi nhiều)
 - Nên explore ít khi agent ĐÃ biết rõ về môi trường (value function ổn định, ít thay đổi)
 - Giải pháp:

$$\begin{aligned} \bullet \quad f(s_t, a_t, \sigma) &= \frac{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}{1 + e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}} = \frac{2}{1 + e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}} - 1 \rightarrow |Q_{t+1}(s,a) - Q_t(s,a)| \text{ càng lớn, } f \text{ càng lớn} \\ \bullet \quad \epsilon_{(t+1)}(s) &= \delta f(s_t, a_t, \sigma) + (1 - \delta)\epsilon_{(t)}(s) \end{aligned}$$

- Sigmoid Epsilon
 - $P_{explore} = 1 - \frac{1}{1 + e^{-\omega|r_1 - r_2|}}$
 - ω : hằng số; $|r_1 - r_2|$: chênh lệch reward của 2 actions
 - $|r_1 - r_2|$ càng lớn $\rightarrow P_{explore}$ càng nhỏ \rightarrow càng explore ít

Câu hỏi

- RL và greedy khác nhau như thế nào?
- Thế nào là chiến thuật epsilon-greedy?
- Khi nào thì nên tăng cường khai thác? khi nào thì nên tăng cường khám phá?

Mục lục

- RL
 - Nguyên lý
 - Đánh giá độ tốt của action
 - Chiến lược lựa chọn action
 - Tabular RL
 - Approximate RL
- DRL
 - Nguyên lý và luồng hoạt động
 - Một số mô hình DRL phổ biến

Tabular RL

- Biểu diễn quan hệ (s, a, q) dưới dạng bảng Q-table
- Cập nhật Q-table sau mỗi action
- Vấn đề của Tabular RL
 - Khi không gian s, a quá lớn \rightarrow kích thước bảng lớn
 - Không xử lý được trường hợp (s, a) hoàn toàn mới
 - Ví dụ: trường hợp huấn luyện model offline

State	Action	Q-value
S_1	A_1	$Q(S_1, A_1)$
S_1	A_2	$Q(S_1, A_2)$
S_m	A_n	$Q(S_m, A_n)$

Approximate RL

Approximate RL

- Biểu diễn quan hệ (s, a, q) thông qua 1 hàm số/model: $\hat{q}(s, a)$
 - Input: vector biểu diễn state, action
 - Các tham số của $\hat{q}(s, a)$ được cập nhật sau mỗi action
- Ưu điểm
 - Không bị ảnh hưởng bởi số lượng state, action
 - Xử lý được trường hợp state, action mới, chưa từng xuất hiện trong quá trình huấn luyện

Supervised learning:

$$\begin{aligned}\hat{q}_w(s, a), q_*(s, a) \\ L = \frac{1}{2} (q_*(s, a) - \hat{q}_w(s, a))^2 \\ w \leftarrow w - \alpha \nabla L\end{aligned}$$

Vấn đề:

Hầu hết các trường hợp,
 $q_*(s, a)$ là không được
biết

Giải pháp:

Ước lượng $q_*(s, a)$

Nhắc lại về Tabular RL

- Tabular Q-learning:
- $$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') \right)$$

$$= Q(s, a) + \alpha \left(\frac{r + \gamma \max_{a'} Q(s', a')}{\text{Giá trị ước lượng: } Y} - \frac{Q(s, a)}{\text{Giá trị thực tế: } X} \right)$$

$$L = \frac{1}{2} \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)^2 \rightarrow \frac{\partial L}{\partial Q} = - \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

$$\rightarrow Q(s, a) = Q(s, a) - \alpha \nabla L \rightarrow \text{Gradient descent}$$

Approximate Q learning

- Tabular Q-learning:

$$\boxed{L = \frac{1}{2} \left(\underbrace{r + \gamma \max_{a'} Q(s', a')}_{Y} - \underbrace{Q(s, a)}_{X} \right)^2 ; Q(s, a) = Q(s, a) - \alpha \nabla L}$$

- Approximate Q-learning: $Q_w(s, a)$

$$\boxed{L = \frac{1}{2} \left(r + \gamma \max_{a'} Q_w(s', a') - Q_w(s, a) \right)^2}$$

- Ví dụ: $Q_w(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$

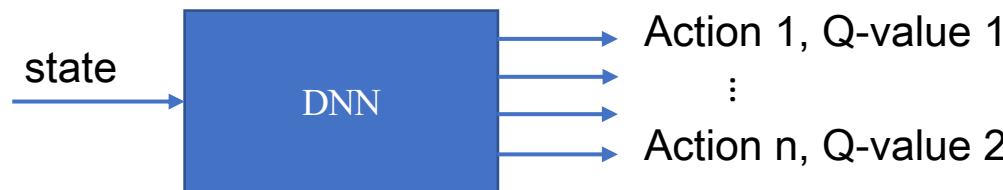
- $w_i \leftarrow w_i + \alpha \left(r + \gamma \max_{a'} Q_w(s', a') - Q_w(s, a) \right) f_i(s, a)$

Approximate RL

- Tổng quát
 - Approximate Q-learning: $Q_{\mathbb{W}}(s, a)$
 - $w_i \leftarrow w_i + \alpha \left(r + \gamma \max_{a'} Q_{\mathbb{W}}(s', a') - Q_{\mathbb{W}}(s, a) \right) \frac{\partial Q_{\mathbb{W}}}{\partial w_i}$
 - SARSA
 - Tabular SARSA: $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha(r + \gamma Q(s', a'))$
 - Approximate SARSA: $w_i \leftarrow w_i + \alpha \left(r + \gamma Q_{\mathbb{W}}(s', a') - Q_{\mathbb{W}}(s, a) \right) \frac{\partial Q_{\mathbb{W}}}{\partial w_i}$

Approximate RL

- $Q_w(s, a)$
 - Có thể là hàm tuyến tính
 - Có thể là 1 ML model: ANN, ...
- DRL
 - Khi $Q_w(s, a)$ là 1 deep learning model \rightarrow DRL



câu hỏi

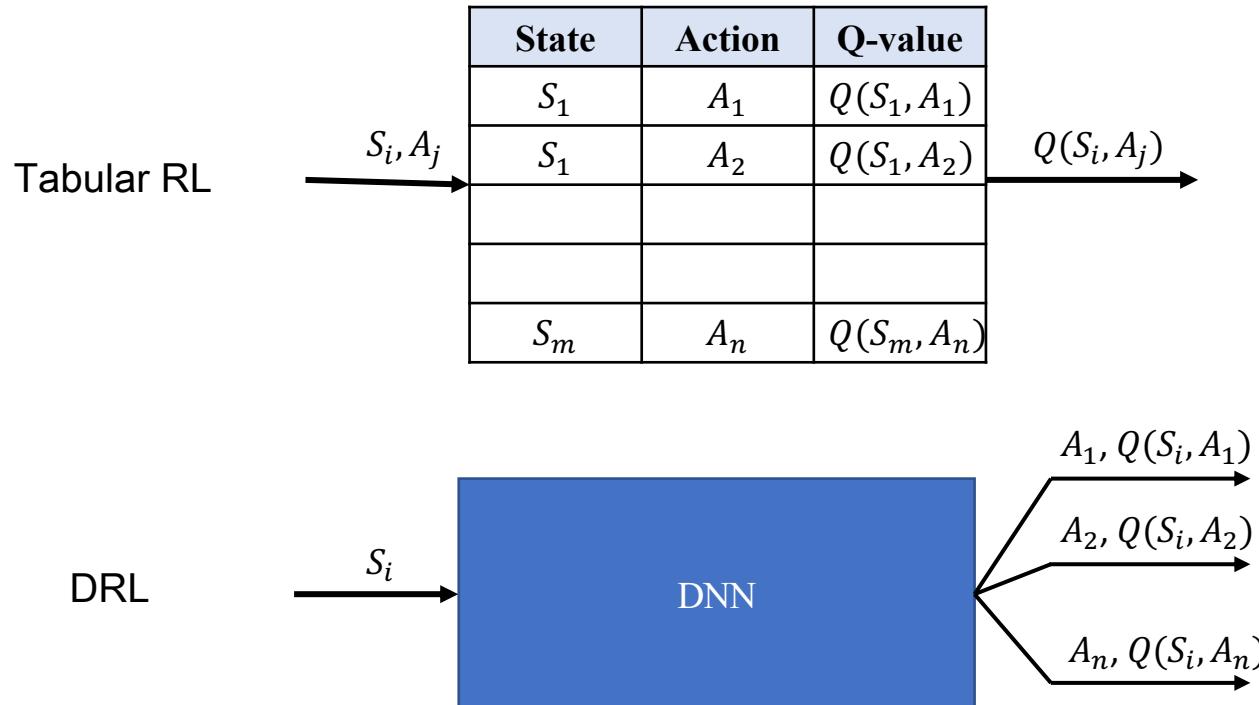
- Nhược điểm của tabular RL là gì?
- Tại sao trong approximate RL lại dùng gradient ascent mà không phải gradient descent?

Deep Reinforcement learning

Mục lục

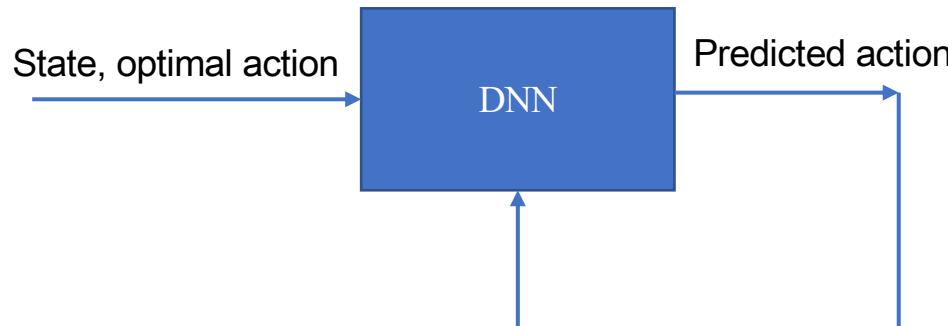
- RL
 - Nguyên lý
 - Đánh giá độ tốt của action
 - Chiến lược lựa chọn action
 - Tabular RL
 - Approximate RL
- DRL
 - Nguyên lý và luồng hoạt động
 - Một số mô hình DRL phổ biến

Tabular RL vs Deep-RL



DRL hoạt động thế nào?

- Off-line training
 - Với những bài toán có sẵn data để huấn luyện: (state, optimal action)



Minimize loss:

$$L = Q(\text{predicted_action}) - Q(\text{optimal_action})$$

DRL hoạt động thế nào?

- Online training:
 - Không biết optimal action
 - Lấy gì làm groundtruth?
 - Công thức Bellman cho chiến lược tối ưu (chiến lược luôn đưa ra action tối ưu)

$$\bullet \quad q_{\pi^*}(s, a) = \mathbb{E}_{\pi^*}[R_{t+1} + \gamma q_{\pi^*}(S_{t+1}, a') | S_t = s, A_t = a]$$

Tìm cách ước lượng giá trị này, và lấy nó làm groundtruth

DQN: sử dụng ý tưởng của Q-learning:

$$\mathbb{E}_{\pi^*}[R_{t+1} + \gamma q_{\pi^*}(S_{t+1}, a') | S_t = s, A_t = a] \approx R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$$

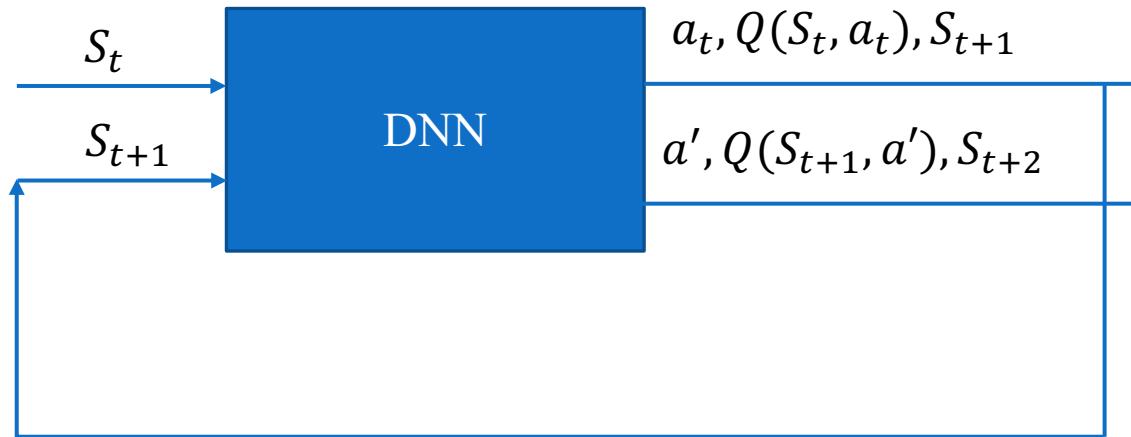
Reward nhận được sau
khi thực hiện action a

Maximum Q value ở trạng thái tiếp theo
(trạng thái sau khi thực hiện xong
action a)

DQN [NIPS 2013]

DQN [NIPS 2013]: sử dụng ý tưởng của Q-learning:

$$\mathbb{E}_{\pi^*}[R_{t+1} + \gamma q_{\pi^*}(S_{t+1}, a') | S_t = s, A_t = a] \approx R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$$



Minimize loss:

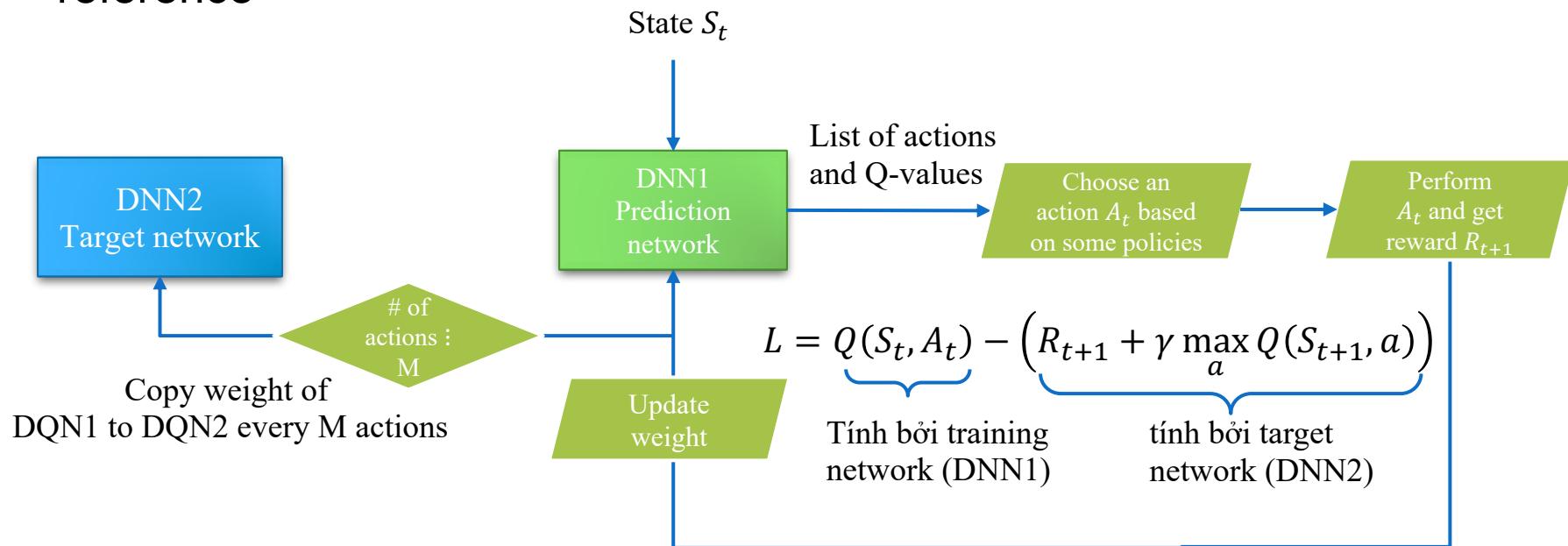
$$L = Q(S_t, a_t) - \left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') \right)$$

Groundtruth thay đổi liên tục
→ Model khó hội tụ

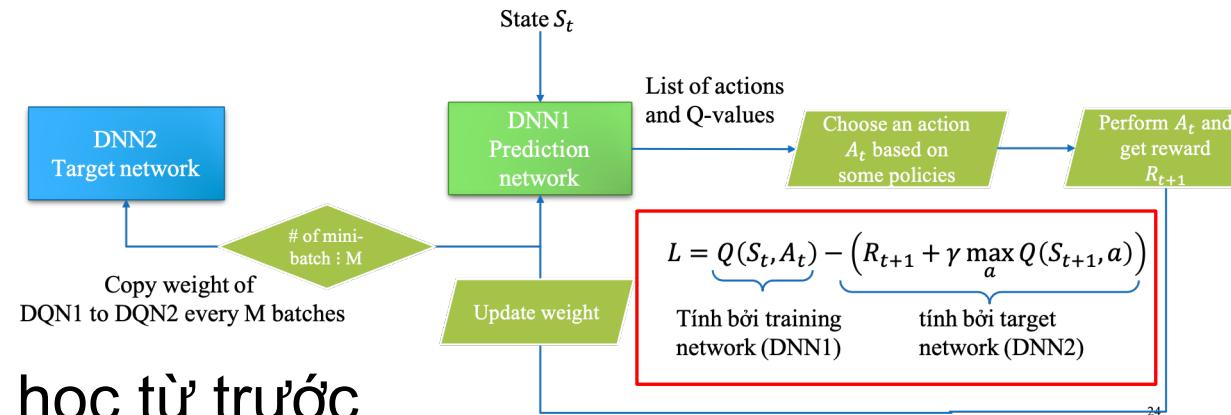
DQN

Ý tưởng: sử dụng 2 mạng DNN

- DNN1: Prediction network (training network): cập nhật weight thường xuyên
- DNN2: Target network: ổn định, ít cập nhật weight → sử dụng như reference



DQN



Vấn đề

- Quên các tri thức đã học từ trước
 - Càng về sau model càng bị quên các tri thức đã học trước đó
- Dữ liệu mất cân bằng
 - Các (state, action) gần nhau có xu hướng giống nhau → mô hình bị mất cân bằng
 - Ví dụ: model chơi game tự động → các scenes gần nhau thường giống nhau

Giải pháp: Experience replay

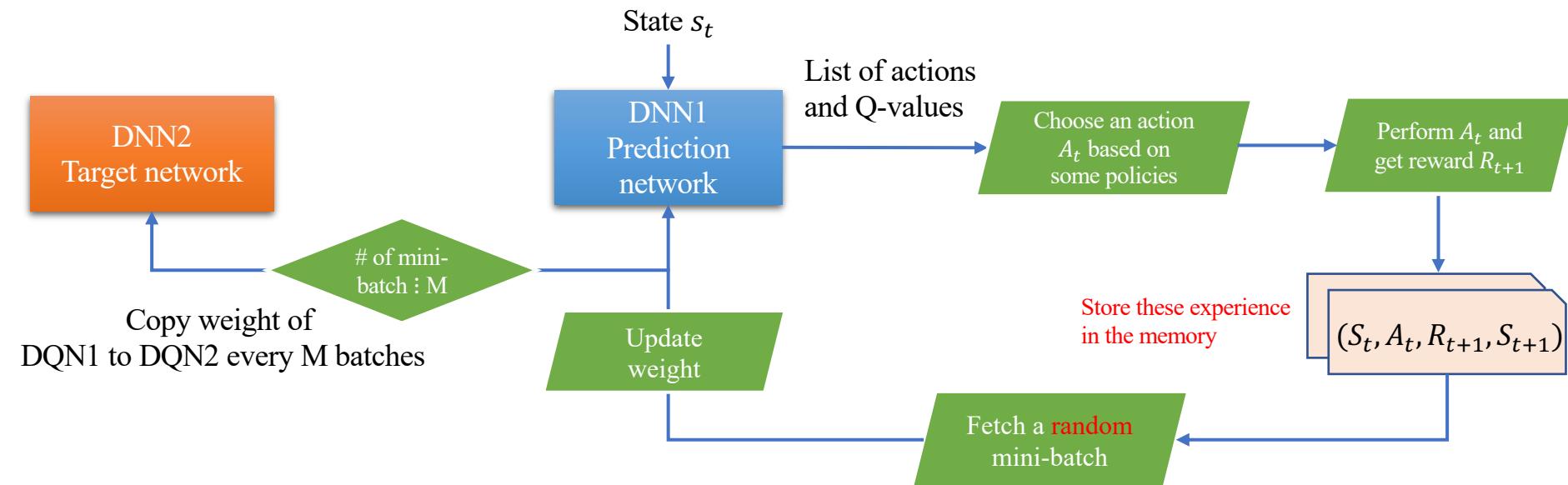
- Lưu tất cả dữ liệu đã từng train vào memory của agent
 - $(S_t, A_t, R_{t+1}, S_{t+1})$
- Định kỳ lấy ra các dữ liệu cũ để huấn luyện model
 - Lấy dữ liệu ngẫu nhiên để tránh correlation

DQN

$$L = \left(Q(S_t, A_t; \theta_1) - \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_2) \right) \right)^2$$

Get from the memory

Recalculated by DNN1
every mini-batch Recalculated by DNN2
every mini-batch



Luồng hoạt động

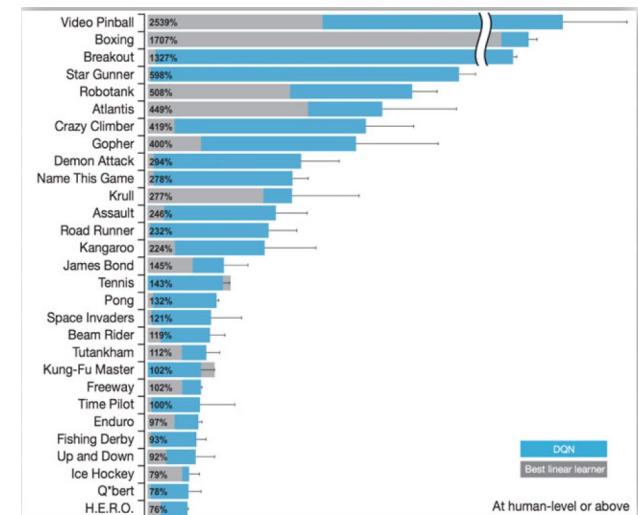
1. Input state S_t vào DNN1 → output là tất cả các actions và các giá trị Q value tương ứng
2. Chọn A_t theo chiến lược
 - Ví dụ $\epsilon - greedy$
3. Thực hiện action A_t , nhận reward R_{t+1} , chuyển sang state mới S_{t+1}
4. Lưu experience $e_t = (S_t, A_t, R_{t+1}, S_{t+1})$ vào memory của agent
5. Chọn mini-batch of experiences từ memory.
 - Với mỗi experience $(S_t, A_t, R_{t+1}, S_{t+1})$
 - Tính $Q(S_t, A_t; \theta_1)$ bằng DNN1
 - Tính $\max_a Q(S_{t+1}, a; \theta_2)$ bằng DNN2
 - Tính loss đối với mini-batch:

$$L_t = \left(Q(S_t, A_t; \theta_1) - \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_2) \right) \right)^2$$

1. Update weight của DNN1 bằng phương pháp gradient descent
2. Sau mỗi M mini-batches, copy weight của DNN1 sang DNN2

Về lịch sử của DQN

- Được đề xuất bởi Google DeepMind
 - V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning", *Nature*, vol. 518, pp. 529-533, Feb. 2015.
- Đánh thắng con người 49 game trong bộ game Atari
- Một số mô hình tương tự DQN cũng được Google dùng trong AlphaGo, phần mềm chơi trò Go



Prioritized experience replay

- Chọn experience dựa trên temporal difference error δ
 - TD error δ càng lớn thì càng được ưu tiên lựa chọn
 - $\delta(V) = R_{t+1} + V(S_{t+1}) - V(S_t)$
 - $\delta(s, a) = R_{t+1} + q(S_{t+1}, A_{t+1}) - q(S_t, A_t)$
 - δ càng lớn \rightarrow thay đổi giữa action value của next state và current state càng lớn
 \rightarrow cơ chế này ưu tiên những experience mang lại nhiều thay đổi
 - $p_i = \delta_i + e$
 - $e > 0$: là hằng số
 - Biến thể
 - $P_i = \frac{p_i^\alpha}{\sum p_k^\alpha}$
 - $\alpha = 0$: xác suất đều
 - α càng lớn \rightarrow ảnh hưởng của p_i càng lớn

Câu hỏi

- Tại sao trong DQN phải dùng hai mạng target và prediction riêng biệt
- Tại sao phải dùng experience replay?

Mục lục

- RL
 - Nguyên lý
 - Đánh giá độ tốt của action
 - Chiến lược lựa chọn action
 - Tabular RL
 - Approximate RL
- DRL
 - Nguyên lý và luồng hoạt động
 - Một số mô hình DRL phổ biến

Double DQN

- Motivation: tránh hiện tượng overestimation
 - Dùng $R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_2)$ để approximate $q_{\pi^*}(s, a) = \mathbb{E}_{\pi^*}[R_{t+1} + \gamma q_{\pi^*}(S_{t+1}, a') | S_t = s, A_t = a]$
- Giải pháp: mượn ý tưởng của Double Q
 - Double Q: dùng 2 Q functions
 - Một Q function dùng để tìm action a có Q max
 - Một Q function dùng để tính Q cho a
 - Double DQN
 - Dùng training network để tìm action a có Q max
 - Dùng target network để tính Q cho a

Double DQN

DQN:

- Target network: tính $\max_a Q(S_{t+1}, a; \theta_2) \rightarrow$ giá trị max này được dùng để update weight
- $L = \left(Q(S_t, A_t; \theta_1) - \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_2) \right) \right)^2$

Double DQN

$$L = \left(Q(S_t, A_t; \theta_1) - \left(R_{t+1} + \gamma \boxed{Q\left(S_{t+1}, \arg \max_a Q(S_{t+1}, A_t; \theta_1); \theta_2\right)} \right)^2 \right)$$

Max Q của mạng DNN2

Action a cho giá trị max của mạng DNN1

Double DQN: propsoed by Google DeepMind, with 3354 citations

Double DQN

- Problem
 - Có thể gây ra hiện tượng underestimate
- Stochastic DQN (IEEE Access 2019)
 - Sử dụng đồng thời cả DQN và Double DQN
 - Gieo 1 biến ngẫu nhiên
 - Nếu biến đấy $> \lambda$: update weight theo cơ chế của DQN
 - Dùng DNN2 để chọn action có Q-value max, dùng Q value max đấy làm tham chiếu update weight
 - Ngược lại, update weight theo cơ chế của Double DQN
 - Dùng DNN1 để chọn Q-value max, dùng DNN2 để tính Q value làm tham chiếu update weight

Dueling DQN

Tách mạng DNN thành 2 nhánh:

- 1 nhánh approximate độ tốt của state
- 1 nhánh approximate mức độ cải thiện của state khi thực hiện hành động

Định nghĩa: advantage function

- $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$

Aggregation

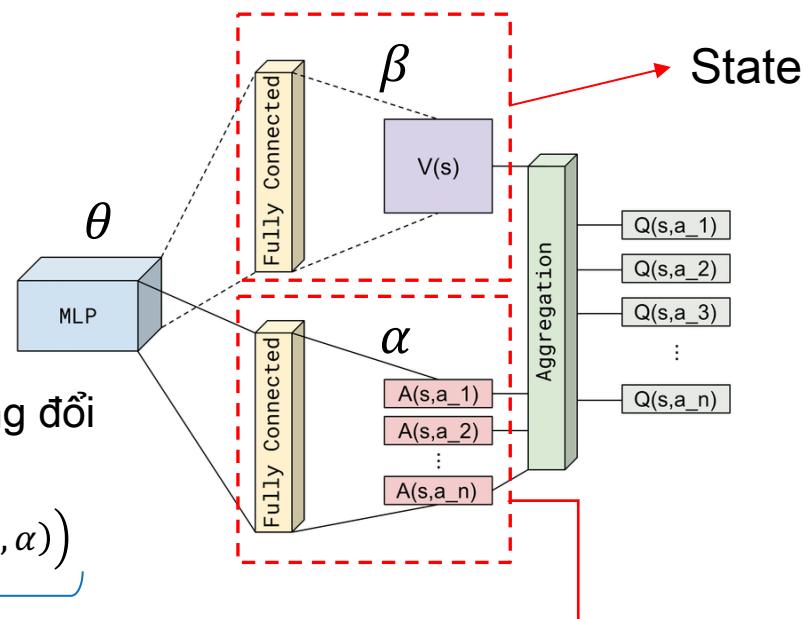
- Nếu $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha)$

- Từ Q không suy ngược được V và A

- Trừ V và cộng A với cùng 1 lượng $\rightarrow Q$ không đổi

- Dùng:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \underbrace{\left(A(s, a; \theta, \alpha) - \max_{a'} A(s, a'; \theta, \alpha) \right)}_{\text{Khi hành động tốt nhất được chọn thì hiệu này} = 0}$$



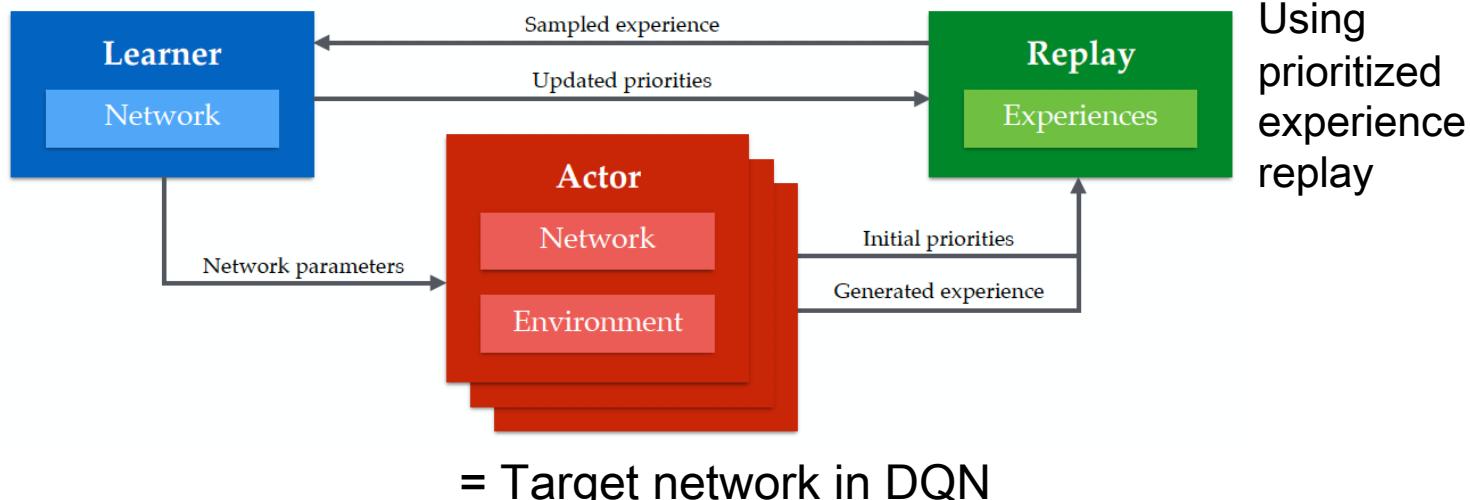
Thực tế: thay $\max_{a'} A(s, a'; \theta, \alpha)$ bằng $\frac{1}{\text{num_of_actions}} \sum_{a'} A(s, a'; \theta, \alpha)$

Ape-X DQN (Distributed prioritized experience replay)

Mục tiêu: Tập trung vào tạo thật nhiều training data

- Key idea 1: sử dụng nhiều mạng actor network (=target network in DQN) để tạo data
 - Mỗi actor network sử dụng 1 environment riêng
 - Data tạo bởi các actor networks được lưu chung vào 1 chỗ
 - Leaner network (=training network in DQN) sử dụng data tạo bởi actor networks để train model
- Key idea 2: sử dụng prioritized experience replay

= Training network in DQN



Coursera

- Practical Reinforcement Learning
- <https://www.coursera.org/learn/practical-rl?action=enroll#enroll>

Policy-based RL

- Monte-carlo
- TD
- SARSA
- Q-learning

Ước lượng value (action value, state value)



Lựa chọn hành động dựa trên value

Value-based RL

Sẽ thế nào nếu chúng ta trực tiếp tối ưu hoá policy (i.e., đưa ra quyết định chọn hành động) thay vì tối ưu hoá việc ước lượng value rồi dựa trên value để quyết định hành động?

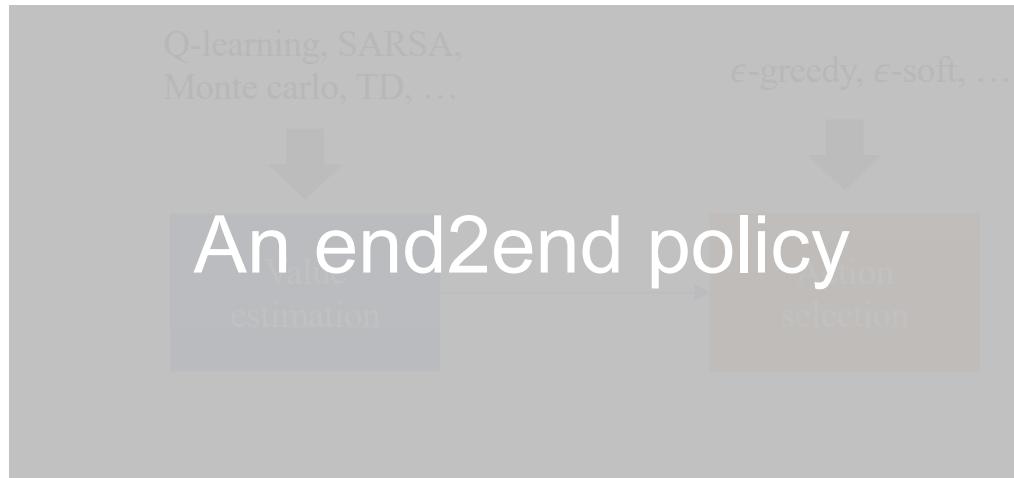


Policy-based RL

Value-based RL vs Policy-based RL



Value-based RL vs Policy-based RL



Policy-based RL

Policy-based RL

▪ Mục tiêu:

- Tìm ra hàm $\pi(a|s)$: xác suất thực hiện hành động a khi ở trạng thái s

▪ Ý tưởng chung

- $\pi(a|s, \Theta)$: π là một hàm số/model, với bộ tham số Θ cần tối ưu
- $J(\Theta)$: hàm đo độ tốt của policy (scalar performance measure)
- $\theta_{t+1} = \theta_t + \alpha \nabla \widehat{J}(\Theta_t) \rightarrow$ gradient ascent method



Ước lượng của gradient của $J(\Theta_t)$

Q1: $J(\Theta_t)$ thường được định nghĩa như thế nào?

Q2: làm thế nào để ước lượng được $\nabla \widehat{J}(\Theta_t)$?

Hàm mục tiêu $J(\Theta)$

- Đối với môi trường hữu hạn (episodic environment)
 - $J(\Theta) = \mathbb{E}_\pi[G_1 = R_1 + \gamma R_2 + \dots + \gamma^{n-1} R_n]$
- Đối với môi trường liên tục (continuing environment)
 - Giá trị kỳ vọng của state value
 - $J(\Theta) = \mathbb{E}_\pi[V(s)] = \sum_s d(s) V(s) = \sum_s \frac{N(s)}{\sum_{s'} N(s')} V(s)$: giá trị kỳ vọng giá trị của tất cả state
 - Giá trị kỳ vọng của reward
 - $J(\Theta) = \mathbb{E}_\pi[r] = \sum_s d(s) \sum_a \pi(a|s, \Theta) R(a, s)$

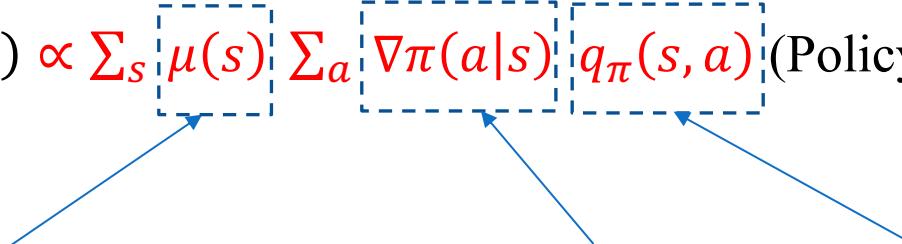
Ước lượng $\widehat{\nabla \mathcal{J}(\Theta_t)}$

- $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla \mathcal{J}(\Theta_t)}$
 - Mục tiêu:
 - Tìm ra một biểu diễn tỉ lệ thuận với $\nabla \mathcal{J}(\Theta) \rightarrow F \propto \nabla \mathcal{J}(\Theta)$
 - Tốt nhất là F là giá trị kỳ vọng của 1 biến e
 - Xấp xỉ F bằng các samples e_t

Ước lượng $\widehat{\nabla \mathcal{J}(\Theta_t)}$

- $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla \mathcal{J}(\Theta_t)}$
 - Mục tiêu:
 - Tìm ra một biểu diễn tỉ lệ thuận với $\nabla \mathcal{J}(\Theta) \rightarrow F \propto \nabla \mathcal{J}(\Theta)$
 - Tốt nhất là F là giá trị kỳ vọng của 1 biến e
 - Xấp xỉ F bằng các samples

Ước lượng $\widehat{\nabla J(\Theta_t)}$ đối với môi trường episode

- $J(\Theta) = \mathbb{E}_\pi[G_1 = R_1 + \gamma R_2 + \dots + \gamma^{n-1} R_n] = v_\pi(s_0)$: state-value tại state đầu tiên
 $\rightarrow v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$
 $\rightarrow \nabla v_\pi(s) = \sum_a \nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \nabla q_\pi(s, a)$
 $\rightarrow \nabla J(\Theta) = \nabla v_\pi(s_0) \propto \sum_s \mu(s) \sum_a [\nabla \pi(a|s) q_\pi(s, a)]$ (Policy Gradient Theorem)

Ước lượng $\widehat{\nabla \mathcal{J}(\Theta_t)}$ đối với môi trường episode

- $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla \mathcal{J}(\Theta_t)}$
- Mục tiêu:
 - Tìm ra một biểu diễn tỉ lệ thuận với $\nabla \mathcal{J}(\Theta) \rightarrow F \propto \nabla \mathcal{J}(\Theta)$
 - Tốt nhất là F là giá trị kỳ vọng của 1 biến e
 - Xấp xỉ F bằng các samples e_t

Ước lượng $\widehat{\nabla J(\Theta_t)}$ đối với môi trường episode

$$\begin{aligned}\nabla J(\Theta) &\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) \\ &= \mathbb{E}_\pi [\sum_a \nabla \pi(a|S_t) q_\pi(S_t, a)]\end{aligned}$$

Xấp xỉ giá trị kỳ vọng bằng các mẫu này

$$\begin{aligned}\widehat{\nabla J(\Theta)} &= \sum_a \nabla \pi(a|S_t) q_\pi(S_t, a) \\ \Theta_{t+1} &\doteq \Theta_t + \alpha \sum_a \nabla \pi(a|S_t) q_\pi(S_t, a) \\ \Theta_{t+1} &\doteq \Theta_t + \alpha \sum_a \nabla \pi(a|S_t, \Theta_t) \hat{q}(S_t, a, \mathbb{w})\end{aligned}$$

REINFORCE algorithm (1992)

REINFORCE = REward Increment = Nonnegative Factor x Offset Reinforcement x Characteristic Eligibility

- Main idea

$$\Theta_{t+1} \doteq \Theta_t + \alpha \sum_a \nabla \pi(a|S_t, \Theta_t) q_\pi(S_t, a)$$

Thay thế tổng này bằng 1 giá trị kỳ vọng \rightarrow xấp xỉ giá trị kỳ vọng bằng mẫu

Tổng trên tất cả action a

\rightarrow muốn chuyển thành giá trị kỳ vọng thì cần có nhân tử $\pi(a|S_t)$

$$\sum_a \nabla \pi(a|S_t) q_\pi(S_t, a) = \sum_a \pi(a|S_t, \Theta_t) q_\pi(S_t, a) \frac{\nabla \pi(a|S_t, \Theta_t)}{\pi(a|S_t, \Theta_t)} \quad \text{Vì } q_\pi(S_t, A_t) = \mathbb{E}_\pi[G_t|S_t, A_t]$$

$$= \mathbb{E}_\pi \left[q_\pi(S_t, A_t) \frac{\nabla \pi(A_t|S_t, \Theta_t)}{\pi(A_t|S_t, \Theta_t)} \right] = \boxed{\mathbb{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t, \Theta_t)}{\pi(A_t|S_t, \Theta_t)} \right]}$$

Xấp xỉ giá trị kỳ vọng $\mathbb{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t, \Theta_t)}{\pi(A_t|S_t, \Theta_t)} \right]$ bằng mẫu $G_t \frac{\nabla \pi(A_t|S_t, \Theta_t)}{\pi(A_t|S_t, \Theta_t)}$

REINFORCE algorithm (1992)

$$\nabla \widehat{J(\Theta)} = G_t \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

$$\rightarrow \Theta_{t+1} \leftarrow \Theta_t + \alpha G_t \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

$G_t \times \nabla \pi(A_t | S_t, \Theta_t)$: nếu G_t càng lớn
thì càng khuyến khích tăng Θ theo
hướng $\nabla \pi(A_t | S_t, \Theta_t)$

Hướng của $\nabla \pi$ làm tăng xác
suất lặp lại hành động A_t
Tức là: nếu tăng Θ một lượng tí
lệ thuận với $\nabla \pi(A_t | S_t, \Theta_t)$ thì
càng ngày A_t sẽ được thực
hiện càng nhiều.

$\frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$: nếu $\pi(A_t | S_t, \Theta_t)$ càng lớn thì

càng không khuyến khích tăng Θ theo
hướng $\nabla \pi(A_t | S_t, \Theta_t)$

→ Tránh trường hợp thực hiện một hành
động quá nhiều lần

REINFORCE algorithm (1992)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Algorithm parameter: step size $\alpha > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|s, \theta)$

Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

$$\begin{aligned} G &\leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \\ \theta &\leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t | S_t, \theta) \end{aligned} \tag{G_t}$$

REINFORCE with Baseline

$$\nabla J(\Theta) \propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a)$$

→Những state có giá trị $q_\pi(s, a)$ quá lớn sẽ chi phối $\nabla J(\Theta)$

→Dùng baseline để normalize $q_\pi(s, a)$

$$\nabla J(\Theta) \propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) (q_\pi(s, a) - b(s))$$

$$\sum_s \mu(s) \sum_a \nabla \pi(a|s) (q_\pi(s, a) - b(s))$$

$$= \sum_s \mu(s) \left[[\sum_a \nabla \pi(a|s) q_\pi(s, a)] - b(s) \sum_a \nabla \pi(a|s) \right]$$

Tiêu chí lựa chọn $b(s)$:

- $b(s)$ có thể là bất cứ hàm nào, miễn là giá trị của nó không phụ thuộc vào a
- Thông thường người ta sẽ chọn $b(s)$ để giảm variance

$$= \nabla \sum_a \pi(a|s) = \nabla 1 = 0$$

REINFORCE with Baseline

$$\Theta_{t+1} \doteq \Theta_t + \alpha G_t \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

$$\rightarrow \Theta_{t+1} \doteq \Theta_t + \alpha(G_t - b(S_t)) G_t \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$



Giá trị kỳ vọng của $v(S_t) \rightarrow 1$ lựa chọn khá tự nhiên cho $b(S_t)$ là giá trị ước tính của $\widehat{v}(S_t)$, $\widehat{v}(S_t, w)$

$$w_{t+1} \doteq w_t + \beta \nabla \widehat{v}(S_t, w_t)$$

REINFORCE with Baseline

REINFORCE with Baseline (episodic), for estimating $\pi_{\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Input: a differentiable state-value function parameterization $\hat{v}(s, w)$

Algorithm parameters: step sizes $\alpha^\theta > 0$, $\alpha^w > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $w \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|s, \theta)$

Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

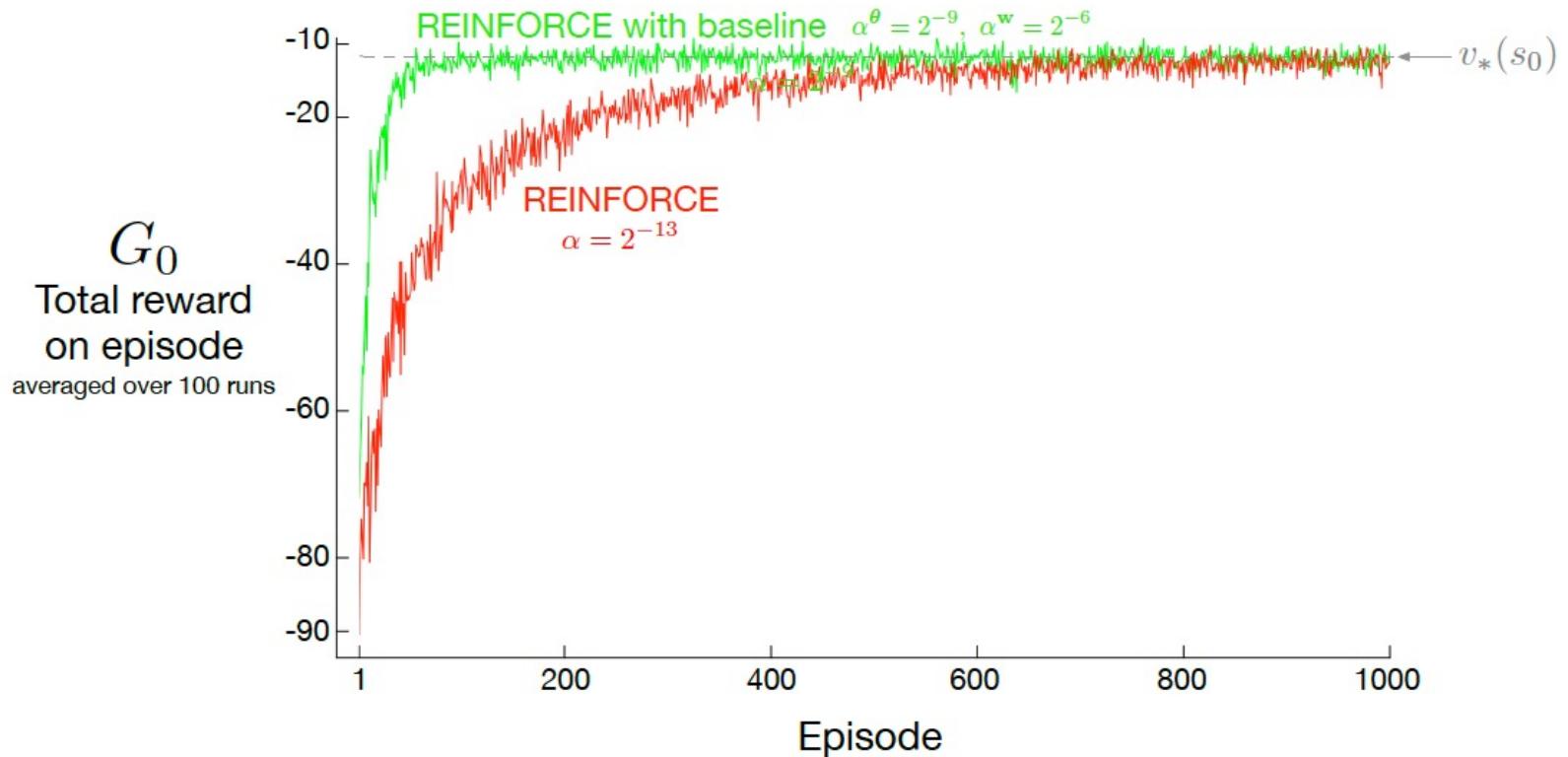
$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \quad (G_t)$$

$$\delta \leftarrow G - \hat{v}(S_t, w)$$

$$w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S_t, w)$$

$$\theta \leftarrow \theta + \alpha^\theta \gamma^t \delta \nabla \ln \pi(A_t | S_t, \theta)$$

REINFORCE (with and without baseline)



Actor-critic learning

- Definition: Methods that learn approximations to both policy and value functions are often called actor-critic methods,
 - Actor: the learned policy
 - Critic: the learned value function
 - Usually a state-value function
- Is policy-based RL with baseline is actor-critic learning?
 - If the baseline is stationary value that never updates with experience → NOT actor-critic
 - If the baseline is estimated from experience → can be called an actor-critic method

One-step Actor–Critic (episodic)

$$\Theta_{t+1} \doteq \Theta_t + \alpha(G_t - b(S_t)) \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

$$\rightarrow \Theta_{t+1} \doteq \Theta_t + \alpha(G_t - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

REINFORCE with baseline

$$R_{t+1} + \gamma \hat{v}(S_{t+1}, w) \leftarrow \text{Actor-Critic learning}$$

$$\rightarrow \Theta_{t+1} \doteq \Theta_t + \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

$$w_{t+1} \doteq w_t + \beta(R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) \nabla \hat{v}(S_t, w_t)$$

Being fully online and incremental

One-step Actor–Critic (episodic)

One-step Actor–Critic (episodic), for estimating $\pi_{\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Input: a differentiable state-value function parameterization $\hat{v}(s, w)$

Parameters: step sizes $\alpha^{\theta} > 0$, $\alpha^w > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $w \in \mathbb{R}^d$ (e.g., to 0)

Loop forever (for each episode):

 Initialize S (first state of episode)

$I \leftarrow 1$

 Loop while S is not terminal (for each time step):

$A \sim \pi(\cdot|S, \theta)$

 Take action A , observe S', R

$\delta \leftarrow R + \gamma \hat{v}(S', w) - \hat{v}(S, w)$ (if S' is terminal, then $\hat{v}(S', w) \doteq 0$)

$w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S, w)$

$\theta \leftarrow \theta + \alpha^{\theta} I \delta \nabla \ln \pi(A|S, \theta)$

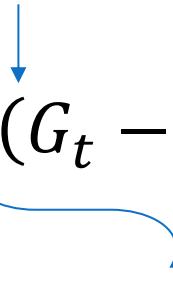
$I \leftarrow \gamma I$

$S \leftarrow S'$

n-step Actor-critic

$G_t = R_t + \gamma R_{t+1} + \dots$: Độ tốt của trạng thái hiện tại S_{t+1}

$$\Theta_{t+1} \doteq \Theta_t + \alpha(G_t - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$



Sự chênh lệch giữa độ tốt của trạng thái tiếp theo và trạng thái hiện tại \rightarrow Advantage

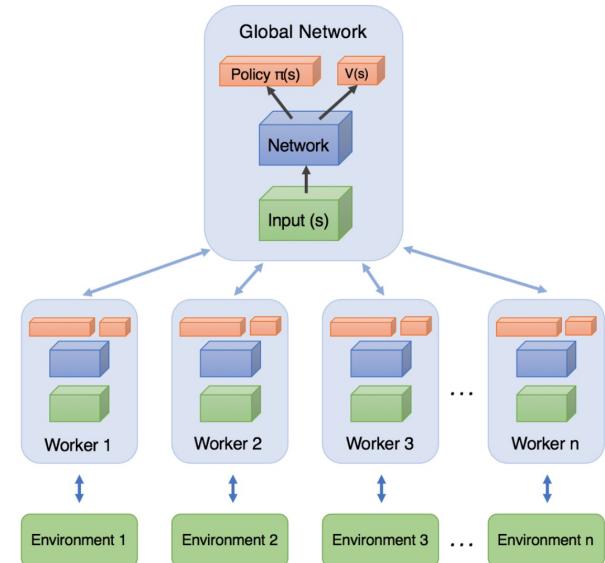
$$\Theta_{t+1} \doteq \Theta_t + \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

$$\begin{aligned} G_t &\sim R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n R_{t+n-1} + \gamma^{n+1} R_{t+n} + \\ &= R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n R_{t+n-1} + \gamma^{n+1} (R_{t+n} + \gamma R_{t+n+1} + \dots) \\ &= R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n R_{t+n-1} + \gamma^{n+1} G_{t+n-1} \\ &\quad \sim R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n R_{t+n-1} + \gamma^{n+1} V(S_{t+n-1}) \end{aligned}$$

$$\Theta_{t+1} \doteq \Theta_t + \alpha(R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n R_{t+n-1} + \gamma^{n+1} \hat{v}(S_{t+n-1}, w) - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t | S_t, \Theta_t)}{\pi(A_t | S_t, \Theta_t)}$$

Asynchronous Advantage Actor-Critic (A3C)

- Multiple independent agents (networks) with their own weights
 - Interact with a different copy of the environment in parallel.
- The agents are trained in parallel and update periodically a global networks
- After each update, the agents resets their parameters to those of the global network



Asynchronous Methods for Deep Reinforcement Learning, NIPS, 2016 (Google DeepMind)

Asynchronous Advantage Actor-Critic (A3C)

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$

// Assume thread-specific parameter vectors θ' and θ'_v

Initialize thread step counter $t \leftarrow 1$

repeat

 Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.

 Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$

$t_{start} = t$

 Get state s_t

repeat

 Perform a_t according to policy $\pi(a_t|s_t; \theta')$

 Receive reward r_t and new state s_{t+1}

$t \leftarrow t + 1$

$T \leftarrow T + 1$

until terminal s_t **or** $t - t_{start} == t_{max}$

$$R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t // \text{Bootstrap from last state} \end{cases}$$

for $i \in \{t - 1, \dots, t_{start}\}$ **do**

$$R \leftarrow r_i + \gamma R$$

Accumulate gradients wrt θ' : $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v))$

Accumulate gradients wrt θ'_v : $d\theta_v \leftarrow d\theta_v + \partial (R - V(s_i; \theta'_v))^2 / \partial \theta'_v$

end for

Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.

until $T > T_{max}$

Copies weights from the global network

Performs t_{max} Samples

$$\begin{aligned} R &= r_{i-1} + \gamma V(s_t, \theta'_v) - V(s_{t-1}, \theta'_v) \\ R &= r_{i-2} + \gamma r_{i-1} + \gamma^2 V(s_t, \theta'_v) - V(s_{t-1}, \theta'_v) \end{aligned}$$

Updates the weights of the global network

Repeats this every T_{max}

Asynchronous Advantage Actor-Critic (A3C)

- To resolve the inconsistency, the global network waits for all the parallel actors to finish their work before updating the global parameters
- In the next iteration the workers start from the same policy
- The synchronized gradient update keeps the training more cohesive and potentially to make convergence faster.



25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**

